AMENDMENTS TO THE CLAIMS

Below is the entire set of pending claims pursuant to 37 C.F.R §1.121(c)(3)(i), with mark-ups showing the changes made in the present Amendment:

1. (Currently amended) A method for presenting data within a computing environment including an application program interface, said method comprising:

creating a universal tagged data object for storing data, the universal tagged data object being a data container that is platform independent, hardware architecture independent, and language independent;

encapsulating a data element into the universal tagged data object to provide universal manipulation and aggregation by computer processing units of tagged data that includes said data element and a corresponding binary tag id; and

packing the tagged data as a binary representation of the tagged data object that is ~~in a machine level code~~ transferable among, and presentable to and ~~directly~~ processible by, without any intermediate data format conversions, any computer processing unit processing data for use by an ~~any~~ application that is prescribed by a data conversion and a formatting specification and that is operating in any computer environment~~, platform, architecture or language~~.

2. (Original) The method as recited in claim 1, wherein packing the tagged data comprises one of the following steps: packing a simple object, packing a complex object, and packing a list object.

3. (Original) The method as recited in claim 2, wherein packing the simple object comprises the steps of:

retrieving a simple object source identifier length;

retrieving a simple object size;

retrieving a simple object type;

retrieving a simple object value;

allocating memory in a packed memory location to accommodate the simple object source identifier length;

copying the simple object size, the simple object type and the simple object value into the packed memory location;

retrieving a simple head value and a simple exit value of the packed memory location; and

copying the simple head value and the simple exit value into the packed memory location.

4. (Original) The method as recited in claim 2, wherein packing the complex object comprises the steps of:

retrieving a complex object source identifier length;

retrieving a complex object field type;

retrieving a complex object field value;

retrieving a complex object field size;

allocating memory in a packed memory location to accommodate the complex object source identifier length;

copying the complex object field type, the complex object field value, and the complex object field size into the packed memory location;

retrieving a complex head value and a complex exit value of the packed memory location; and

copying the complex head value and the complex exit value into the packed memory location.

5. (Original) The method as recited in claim 4, wherein the complex object field value is a simple object.

6. (Original) The method as recited in claim 2, wherein packing the list object comprises the steps of:

retrieving a list object source identifier length;

allocating memory in a packed memory location to accommodate the list object source identifier length;

retrieving a list object array;

copying the list object array into the packed memory location;

retrieving a list head value and a list exit value of the packed memory location; and

copying the list head value and the list exit value into the packed memory location.

7. (Original) The method as recited in claim 6, wherein the list object array comprises a simple object and a complex object.

8. (Canceled)

9. (Original) The method as recited in claim l, wherein the method further includes the step of transmitting the binary representation of the tagged data.

10. (Original) The method as recited in claim 1, wherein the tagged data object comprises one of a simple data object, a complex data object, and a list data object.

11. (Original) The method as recited in claim 10, wherein the simple data object comprises a simple data wrap around.

12. (Original) The method as recited in claim 10, wherein the complex data object comprises a named tree including a data storage having a field name connected with a value.

13. (Original) The method as recited in claim 10, wherein the list data object comprises a combination of the simple data object and the complex data object.

14. (Original) The method as recited in claim 1, wherein encapsulating the data element comprises the steps of:

determining a type of data to be encapsulated;

associating said type of data to the corresponding tag id; and

writing said corresponding tag id and said data element into the tagged data object.

15. (Original) The method as recited in claim 14, wherein the type of data comprises one of the following: an integer, a float numeric value, a one byte value, a character string, a zero terminated character sequence, a byte sequence, a binary data, a null value, a java object, a TD object, an XML text object, a primitive data type, a compound data type, and a list data type having a combination of data types.

16. (Original) The method as recited in claim 14, wherein the corresponding tag id comprises one of the following: short, ushort, long, ulong, float, double, byte, cstring, blob, null, llong, longstr, java_object, object and xmlstr.

17. (Original) The method as recited in claim 1, wherein the method further comprises the step of determining a number of data sequences to be tagged.

18. (Original) The method as recited in claim 1, wherein encapsulating the data element comprises the step of adding to the tagged data object the following: a data, a position and a tag data element.

19. (Original) The method as recited in claim 1, wherein prior to packing the tagged data, the tagged data object is transformed from a first type to a second type to provide for a change in properties.

20. (Original) The method as recited in claim 19, wherein the first type comprises one of a simple object, a complex object, a list object, a multiplicity of simple objects, a multiplicity of

complex objects and a multiplicity of list objects.

21. (Original) The method as recited in claim 19, wherein the second type comprises one of a simple object, a complex object, a list object, a multiplicity of simple objects, a multiplicity of complex objects and a multiplicity of list objects.

22 - 42 (Canceled)

43. (Currently amended) A method for presenting data within a computing environment of the type having an application program interface prescribed by a data conversion and a wire formatting specification, said method comprising the steps of:

creating a tagged data object, wherein the tagged data object comprises a universal data container that is platform independent, hardware architecture independent and language independent, said tagged data object to provide universal manipulation and aggregation by computer processing units of a structured data and an unstructured data;

encapsulating a data element into the tagged data object to provide tagged data that includes said data element and a corresponding binary tag id;

providing a tagged data transmission by packing the tagged data as a binary representation of the tagged data object that is ~~in a machine level code~~ transferable among, and presentable to and ~~directly~~ processible by, without any intermediate format conversions, any computer processing unit processing data for use by an ~~any~~ application that is prescribed by a data conversion and formatting specification and that is operating in any computer environment~~, platform, architecture or language~~;

transmitting the tagged data transmission;

unpacking the tagged data from the binary representation; and

extracting the data element from the tagged data.

44. (Original) The method as recited in claim 43, wherein packing the tagged data comprises one of the following steps: packing a simple object, packing a complex object, and packing a list object.

45. (Original) The method as recited in claim 44, wherein packing the simple object comprises the steps of.

retrieving a simple object source identifier length;

retrieving a simple object size;

retrieving a simple object type;

retrieving a simple object value;

allocating memory in a packed memory location to accommodate the simple object source identifier length;

copying the simple object size, the simple object type and the simple object value into the packed memory location;

retrieving a simple head value and a simple exit value of the packed memory location; and

copying the simple head value and the simple exit value into the packed memory location.

46. (Original) The method as recited in claim 44, wherein packing the complex object comprises the steps of:

retrieving a complex object source identifier length;

retrieving a complex object field type;

retrieving a complex object field value; retrieving a complex object field size;

allocating memory in a packed memory location to accommodate the complex object source identifier length;

copying the complex object field type, the complex object field value, and the complex object field size into the packed memory location;

retrieving a complex head value and a complex exit value of the packed memory location; and

copying the complex head value and the complex exit value into the packed memory location.

47. (Original) The method as recited in claim 46, wherein the complex object field value comprises a simple object.

48. (Original) The method as recited in claim 44, wherein packing the list object comprises the steps of:

retrieving a list-object source identifier length;

allocating memory in a packed memory location to accommodate the list object source identifier length;

retrieving a list object array;

copying the list object array into the packed memory location;

retrieving a list head value and a list exit value of the packed memory location; and

copying the list head value and the list exit value into the packed memory location.

49. (Original) The method as recited in claim 48, wherein the list object array comprises a simple object and a complex object.

50. (Original) The method as recited in claim 43, wherein unpacking the packed tagged data comprises one of the following steps: unpacking a simple object, unpacking a complex object, and unpacking a list object.

51. (Original) The method as recited in claim 50, wherein unpacking the simple object comprises the steps of:

retrieving a simple head value and a simple exit value to provide a simple object source identifier length;

allocating memory in an unpacked memory location to accommodate the

simple object source length; and

copying a simple object size, a simple object type and a simple object value into the unpacked memory location.

52. (Original) The method as recited in claim 50, wherein unpacking the complex object comprises the steps of:

retrieving a complex head value and a complex exit value to provide a complex object source identifier length;

allocating memory in an unpacked memory location to accommodate the complex object source length; and

copying a complex object field type, a complex object field value and a complex object field size into the unpacked memory location.

53. (Original) The method as recited in claim 52, wherein the complex object field value comprises a simple object.

54. (Original) The method as recited in claim 50, wherein unpacking the list object comprises the steps of:

retrieving a list head value and a .list exit value to provide a list object source identifier length;

allocating memory in an unpacked memory location to accommodate the list object source length; and

copying a list object array into the unpacked memory location.

55. (Original) The method as recited in claim 54, wherein the list object array comprises a simple object and a complex object.

56. (Original) The method as recited in claim 43, wherein the tagged data object comprises one of a simple data object, a complex data object, and a list data object.

57. (Original) The method as recited in claim 56, wherein the simple data object comprises a simple data wrap around.

58. (Original) The method as recited in claim 56, wherein the complex data object comprises a named tree including a data storage having a field name connected with a value.

59. (Original) The method as recited in claim 56, wherein the list data object comprises a combination of the simple data object and the complex data object.

60. (Original) The method as recited in claim 43, wherein encapsulating the data element comprises the steps of:

determining a type of data to be encapsulated;

associating said type of data to the corresponding tag id; and

writing said corresponding tag id and said data element into the tagged data object.

61. (Original) The method as recited in claim 60, wherein the type of data comprises one of the following: an integer, a float numeric value, a one byte value, a character string, a zero terminated character sequence, a byte sequence, a binary data, a null value, a java object, a TD object, an XML text object, a primitive data type, a compound data type, and a list data type having a combination of data types.

62. (Original) The method as recited in claim 60, wherein the corresponding tag id comprises one of the following: short, ushort, long, ulong, float, double, byte, cstring, blob, null, llong, longstr, java_object, object and xmlstr.

63. (Original) The method as recited in claim 43, wherein the method further comprises the step of determining a number of data sequences to be tagged.

64. (Original) The method as recited in claim 43, wherein encapsulating the data element comprises the step of adding a data, a position and a tag data element to the tagged data object.

65. (Original) The method as recited in claim 43, wherein the tagged data object is transformed from a first type to a second type to provide for a change in properties.

66. (Original) The method as recited in claim 65, wherein the first type comprises one of a simple object, a complex object, a list object, a multiplicity of simple objects, a multiplicity of complex objects and a multiplicity of list objects.

67. (Original) The method as recited in claim 65, wherein the second type comprises one of a simple object, a complex object, a list object, a multiplicity of simple objects, a multiplicity of complex objects and a multiplicity of list objects.

68. (Original) The method as recited in claim 43, wherein extracting the data element from the tagged data comprises the steps of:

determining the type of said data element to provide a corresponding tag id; and

writing the data element into the tagged data object.

69. (Original) The method as recited in claim 68, wherein the type of data comprises one of the following: an integer, a float numeric value, a one byte value, a character string, a zero terminated character sequence, a byte sequence, a binary data, a null value, a java object, a TD object, an XML text object, a primitive data type, a compound data type, and a list data type having a combination of data types.

70. (Original) The method as recited in claim 68, wherein the corresponding tag id comprises of one of the following: short, ushort, long, ulong, float, double, byte, cstring, blob, null, llong, longstr, java_object, object and xmlstr.

71. (Original) The method as recited in claim 68, wherein writing the data element into the tagged data object comprises the step of adding a data, a position and a tag data element to the tagged data object.